Congestion Control through Load Balancing Technique for Mobile Networks: A Cluster based Approach

Subhajit Sadhu[#1], Sushovon Maity[#2],
[#] Department of CSE, NIT, Durgapur, WB-713209, INDIA
[1] subhajitsadhu.nitdgp@gmail.com, [2]sushovon.nit@gmail.com

Abstract — **The Optimal Routing Path (ORP) for mobile cellular networks is proposed in this paper with the introduction of cluster-based approach. Here an improved dynamic selection procedure is used to elect cluster head. The cluster head is only responsible for the computation of least congested path. Hence the delay is reduced with the significant reduction on the number of backtrackings.**

Keywords - **clustering, load-balancing, congestion control, computation optimization, delay reduction.**

I. Introduction

Allocation of resources and optimization of its usage are the major concerns in the field of mobile networking due to the viral increase of mobile phone usage. Sometimes calls have to be blocked or dropped due to unavailability of resources. Quality of Services (QoS) may also degrade, leading to customer dissatisfaction. Therefore, minimizing the number of calls blocked per unit time, along with providing better quality of services (QoS) with optimum delay, is being considered as one of the primary focus of research. Hence different resource allocation schemes have been proposed for better user services and system utilization [1].

One of those schemes is based on the concept of clustering of the network nodes [2]. It tries to minimize the total computational overhead and delay by allowing only the cluster-heads to compute the least congested optimal routing path through that particular cluster; thereby ensuring not only congestion control but also load balancing. But in this approach, the cluster head becomes overly loaded with computation of the Optimal Routing Path (ORP) and hence thereby, performance may decrease and delay may increase at the time of heavy traffics. Another one of those scheme based on the concept of dynamic pricing [2] has been proposed which introduces variant price for calls in accordance with variation of demands of the users, but it does not elucidate the problem caused by computational overhead. Effective joint Call Admission Control (JCAC) [3] algorithms have been proposed to ensure optimal QoS. The reduction of delay was beyond the scope of this work. Another dynamic pricing scheme introduced as priority based tree generation model (PTGM) in [1] proposing the use of a fixed MOBILE TERMINAL (MT) acting as a mobile switching centre (MSC). Further, the improvement on foundation of optimal path is needed. The reduction of routing path was dealt with the introduction of certain imperative rules based on the radii oriented topology of nodes and well-defined priority factors [4]. The probable chances of backtracking increased the delay to a larger extent.

In this paper, the previous work in [1, 2, 4] has been extended with improved dynamic selection procedure to, select the appropriate node as autonomous cluster-head, in order to distribute the computation overhead equally among that particular subset of nodes belonging to that cluster. The cluster head is only involved in the computation of Optimal routing path (ORP). Hence the objective of this work is to search the ORP through intermediate cellular clusters with least backtracking, and also to change the autonomous cluster-head dynamically, to distribute the load of computation.

The paper is organized as follows: section 2 describes the proposed model along with its working principles. The experimental results have been discussed in section 3.Finally, section 4 contains the conclusion.

II. Proposed Model

The model addressed in this work is based on the cellular layout of mobile network shown in Fig.1.

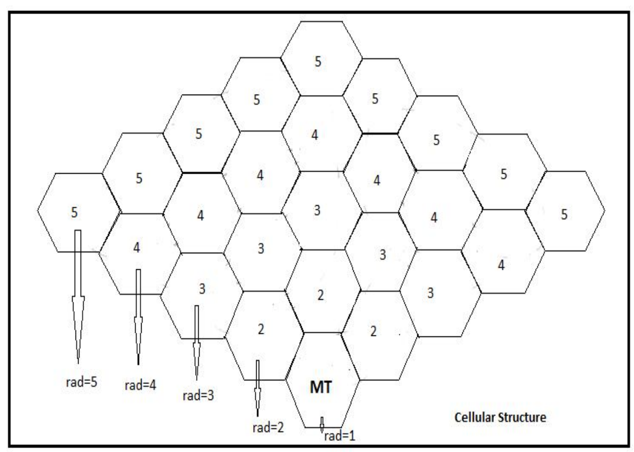

Fig. 1: Cellular layout of mobile networks

The calls are forwarded through the MT [1]. The proposed model is based on the functional activities described in the following.

A. Clustering Model

Firstly the nodes are assembled in the form of a graph. The nodes are grouped into various levels, starting from level 1 where, the MT [1] is placed. Now the nodes are grouped into various clusters[6]. The condition for cluster formation is discussed later. The transmission of signal from MT to destination node takes place from cluster to cluster.

DOI: 01.IJNS.4.1.1160

ACEEE Int. J. on Network Security , Vol. 4, No. 1, July 2013

Introduction of Index Structure:

During the call transmission between nodes, each cluster keeps on computing the least congested ORP through it, towards its upper radius. Here, we are not able to determine whether this computed path actually leads to the destination node. If not then it would cause huge futile computation. Subsequently it requires back-tracking which would increase computational complexity and delay. Hence an Index structure has been introduced to reduce backtracking. The method of indexing is described below:-

We index the nodes of the graph in such a way that the index number of the left child is less than that of its parent node and the right child has the index number greater than the parent. Thus intermediate nodes are given indices which lie between these extreme child indices. So, the cluster head forwards the call, only if the destination index number lies between those two extremities. Otherwise, the call is blocked through that specific node. Some cases might arise, where the two different nodes belonging to different radii are assigned with same indices. To overcome such ambiguity we couple radius number (rad_i) with node index (I_i) ,and this conjugation serves as the unique identification (u_{id}) for each node. This method is described by the following algorithm:

```
Initialize (MT,n,r) {       /* Pre_Computation */
 for (i=1 to n)
   {
      S_i ← U_m;          // Um stands for status "unmarked"
      A_i ← Φ;            // Acceptance Rate of (i);
      R_i ← Φ;           //Rejection Rate of (i);
      L_i ← Φ;           // Load of (i);
   }
  I_MT = n/2;   Ident_no(MT,n/2,1);
}
/*assigning unique identification number to each node*/
Ident_no(node,m,r)  {
   if (node==NULL)
    Return;
   if (lc_node!=NULL)
       I_lc_node ← m-1;  rad_lc_node ← r+1;
   Ident_no (lc_node, m-1,r+1);
   if (rc_node!=NULL)
       I_rc_node ← m+1;  rad_lc_node ← r+1;
   Ident_no (rc_node, m+1, r+1);
 }  /*composition of {rad_i, I_i} serves as the u_id */
Extremity_indexing(n,r,k) {    /* Extremity Calculation */
 for (i=1 to k)
   E_Cal(ni,ni,ni->lc,ni->rc);
E_Cal(node,temp,temp1,temp2) {
  If (temp! =NULL)  {
    temp1 ← node->lc;  temp2 ← node->rc;
    Call E_Call(node,node->lc,templ,temp2);
    node->LeftExtreme ← I_templ;
    Call E_Call(node,node->rc,templ,temp2);
    node->RightExtreme ← I_temp2;
   }
 }
```

This algorithm is verified with the following Fig. 2. Let the unique id of the destination node be d_{uid} . Now the call is forwarded if $5{:}N/2-4 <= d_{uid} <= 5{:}N/2+4$ is satisfied else blocked.

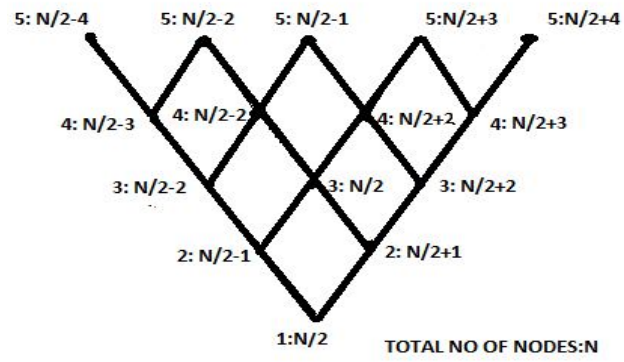

Fig.2: u_{id} assignment to nodes

Cluster Formation

The number of nodes increases exponentially with the increment of radii. The cluster size is dependent on the number of nodes at the specific level. To maintain a trade off between the number of clusters and the cluster cardinality, the cluster size is considered as variable and is considered to be a function of the respective level number. The size is defined as: $(2^l + 1)$ or $(2^l - 1)$ depending on the remaining number of nodes where l=level number. We have chosen $(2^l + 1)$ or $(2^l - 1)$ such size cluster because the number of nodes in a complete binary tree with height h is 2^h-1[7].

Grouping of nodes into various clusters

Obviously, the number of nodes in (n+1) level is greater than that in n^{th} level. Here, one-to-many computation requires less number of computations than one-to-one in model [5].

Static Selection of cluster head (C_i)

In each cluster[6] a node is selected as the cluster head which is the lowest index from the lower radius. This is used to maintain uniformity

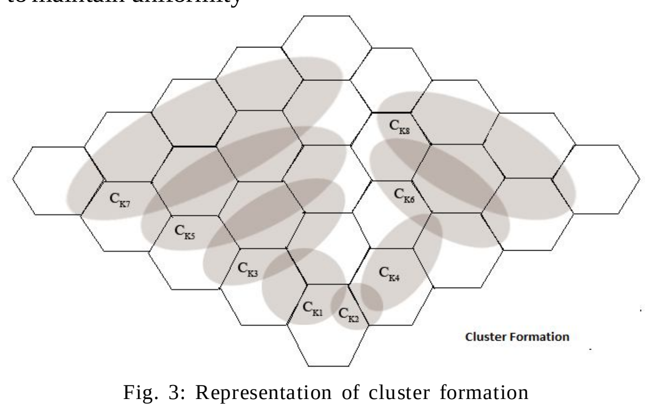

Fig. 3: Representation of cluster formation

The cluster formation based on the discussed algorithm is shown in Fig. 3.

Dynamic Selection of cluster head (C_i)

The cluster head besides taking decisions regarding call transmission, also itself takes part in call transmission. It may sometimes happen that the cluster head may become too busy in call transmission, if its position lies on an unavoidable busy route. In this situation the call transmission decisions

DOI: 01.IJNS.4.1.1160

may get pending, resulting in delay in transmission, as the cluster head is in non- pre-emptive call transmission through itself.

The solution is instead of keeping the cluster head static, we have to dynamically change the cluster head at runtime when required. Under this, any node in a cluster may be assigned the responsibility of cluster head, if it happens to be the idlest node of the cluster at that point of time.

A Threshold Load value (L_T) will be calculated for each cluster, which is defined as the ratio of, summation of load of all notes in that cluster, to that of the size of the cluster.

L_{Tk} = “ (L_i) / (S_k)

L_{Tk} = “ (A_i-R_i) / (S_k)

Where, L_i is the load of node i, S_k is the size of K^{th} cluster and L_{Tk} is the Threshold Load value for K^{th} cluster.

Whenever the load of the cluster head (L_k) will be more than Threshold Load value of the cluster (L_{Tk}), then a change of cluster head procedure will be initiated.

So, when (L_k>L_{Tk}), the node in that cluster with least load value will be made the new cluster head.

C_k = Node with Min {L_i} in K^{th} cluster.

Where C_k is the new cluster head of K^{th} cluster and L_i is the load of i^{th} node in K^{th} Cluster.

This shifting of cluster head will be accompanied by shifting of the entire data set containing information about that cluster from the previous cluster head to presently selected cluster head. Also the priorities of the nodes need to be swapped between the current and the previous cluster head, in order to keep the current cluster head free as much as possible. So the new priorities(p) will be:

p = 1/α i.e. priority for previous cluster head

p = α i.e. priority for current cluster head,

Where, 0 <= α <= 1 (α: constant) & p is the priority of the nodes. (Discussed in section *2.3)*

Since under this procedure any node in a cluster can be a cluster head, so an array data structure Cluster_Head[i] is used to store the cluster head index, to keep a track of all the cluster heads. The Cluster_Head[i], will store the index of cluster head of the i^{th} cluster. Thus the shifting of cluster head will require modifying this data also.

This entire procedure of calculating the total load of the cluster and swapping the cluster head if required will take place in each and every cluster from time to time after a certain timer interrupt T_R (constant).

So, initially the cluster head will be statically selected and later during runtime the cluster head may change depending upon network traffic (load).

This method is described by the following Pseudo Code:

Dynamic_Cluster_Selection(Cluster i)

```
{
set Li=0;
for (j=1 to last node in ith cluster) {
   Li+=(Aj-Rj);        //cluster load summation
}
LTi=(Li / (Si);   //where, Si is size of cluster i,
                  // LTi is the Threshold Load value for ith cluster.
Ci= Cluster_Head[i];  // cluster head of ith cluster
if (Load (Ci) < LTi)
{    //load on current cluster head is less than threshold load
return; // No change in cluster head is required
}
else //load on current cluster head is greater than threshold
{    //change in cluster head
 min= Load (Ci);
 node=i;
 for (j=1 to last node in ith cluster)
 {
   if (Lj<min)
   {
    Min= Lj;
    Node=j;
    }
  }
  swap_priority(Ci,j); // swapping priorities of nodes Ci & j
  TransferOfClusterData (node i → node j);
  Cluster_Head[i]=j; // cluster head of ith cluster → jth node
 }
return;
}
```

B. Function of the cluster head

The cluster head is an autonomous node which keeps a track of detailed information of each and every node belonging to that particular cluster. Each node has the following information embedded in its structure:

(a)Call Acceptance Rate (A_i): no of calls accepted by node n_i per unit time.(b)Call Rejection Rate (R_i): no of calls dropped by node n_i per unit time.(c)Priority ($\div_i$): Priority of the node (defined later). (d)Left Extremity (Le_i): Left Extreme node reachable form node i. (e) Right Extremity (Re_i): Right Extreme node reachable form node i, Unique id (u_{id}), Radius (rad_i) and Load (L_i) =A_i -R_i.

Besides, the cluster head keeps the information on connectivity of the clustered network. The entire computation of candidate ORPs and thereby selecting the optimum one is carried out by the cluster-head only. So cluster head is solely responsible for transmitting the incoming request from lower radius to the appropriate node in higher radius

C. Priority of the nodes

The main computations of candidate ORP are being carried out explicitly by the cluster heads. So it is imperative to make the cluster head less burdened that is to make sure that less number of calls physically passes through cluster heads only. In order to implement this, we introduce priority among the nodes belonging to a cluster, so that low priority nodes (nodes other than heads) transmit calls. Whereas high priority nodes (cluster heads) are employed to computation mainly. The priority of a node is defined as follows:-

p = 1/á for nodes other than cluster head

p = á for cluster head,   where   0 <= á <= 1 (á: constant)

The desirable performance can be achieved by assigning appropriate value to á. For high transmission á should be set near to 1, for steady performance á should be set near to 0.5

ACEEE Int. J. on Network Security , Vol. 4, No. 1, July 2013

and for restricted use á should be set near to 0.

D. Load balancing by cluster head

Whenever a call request arrives at a node, the corresponding cluster head computes the cost of congestion for each possible path and to choose the least congested one. The Cost of Congestion (CoC) of the Routing path (£) is expressed as:

£= 1/ (Φ)

Where, Φ= (Load of node) * (Priority of node) = L* p;

= (A-R) * p;

Therefore, ORP= min {$£_i$} for all i.

E.Call transmission procedure

Initially the call request comes to the MT. The cluster head (C_i) to which MT belongs reads the uid of the destination node. It then reads the Re_i and Le_i of MT and if Le_i<=uid<=Re_i then computation proceeds else the call is dropped instantly by that cluster head.

If the above said criterion is satisfied then least cost ORP is calculated by the cluster head in the following way:

First all the candidate paths from the lower radius (rad_i), to the immediate higher radius (rad_i +1) are computed, which proceeds in the direction of the destination. Then the cost of each path is computed according to the previous discussion.

$£_i$ =1/ [(A_i-R_i) * p_i]

Here, i denote the node in radius (rad_i+ 1) adjacent to MT node present in the same cluster C_i.

The cluster head C_i selects the $£_i$ with least cost and forwards the incoming request from MT. This process continues until the ultimate destination node is reached or unless the call gets blocked by any cluster head. This procedure is described by the following algorithm.

```
Algorithm:    /* Selection of ORP */
Path_Selection (MT, destination_node) {
Source_nodeßMT;   Countß0;  Set int_node    ! 0;
While (destination_node!=int_node or Count<=Max)  {
   C_head ßCluster_head of Source_node;
   Flagßforwarding (C_head,Source_node,destination_node);
   If (Flag=-1) {
     Block_Request();   break;
     }
    Int_node=Flag;
  }
}
Forwarding (C_head, Source_node, destination_node) {
 If (Le Source_node <= destination_node <= Re Source_node)
 {    Max=0;
 For (all nodes i in next radius in that cluster adjacent to Source_node)
  Cost = (A_i-R_i)* ÷_i;
  If (max<Cost) {
   Max=Cost;    Node =i;
   }
If (Max==0)
   Return -1;
   Ai++;    Return (i);
   }
   Else
    Return (-1);
   }
```

F. Link state routing protocol

In each cluster the cluster head keeps the track of different information of the nodes in the cluster discussed previously. As the transmission continues, the acceptance and rejection of the nodes keep on changing depending on traffic. The priorities of the nodes may also be changed explicitly due to changing demand or due to swapping to cluster head. The updated information has to be transmitted to every cluster head of the network which will then update the information locally among the nodes in that cluster. So the base cluster head containing the MT is provided with updated link state packet (ULSP) which keeps this update information. This LSP gets flooded to all the neighbouring cluster heads in higher radius except the one from which it has received. The cluster head keeps this LSP if it's the updated version of the previous LSP else it keeps the previous one by checking the reference number associated with each LSP. This flooding of LSP will take place after a certain time interval (t_m).So the flooding stops when every cluster head retains the updated LSP.

G. Delay Computation

In order to compare the performance and speed of the proposed model with those of the previous model, we need to compute the overall delay while transmission through MT to destination node.

The overall delay is a sum of two types of delay. These are:-

(a)Propagation Delay: Delay incurred due to traversal of the physical topology. So the parameter on which propagation delay depends is the number of physical hops traversed.

Let the number of nodes from MT to destination node be n. So the total number of hops traversed is log(n). Hence Propagation Delay is (h**d*) where d is propagation delay per edge.

(b) Computation Delay: Delay incurred due to computation analysis by the cluster head. Hence Computation delay is (h**k*) where k is computation delay per node.

Hence, overall delay: Propagation Delay + Computation Delay.

$$\textbf{DELAY } (D) = [\, \{\mathrm{Log}\,(n)*d\} \;+\; [\,\{\mathrm{Log}\,(n)-1\}*\,k]\,]$$

Delay incurred when the cluster-head remains static:

As discussed in section *2.1.5* when the cluster head becomes busy in call transmission, the computation and decisions to be taken remain pending as long as the cluster head is busy with its transmission. This will lead to an additional delay which can be termed as Waiting Time Delay (WTD) and can be defined as follows:

WTD=f(L_i,T_t)

Where f () is a variable function of L_i and T_t and will vary from network to network.

DOI: 01.IJNS.4.1.1160

ACEEE Int. J. on Network Security , Vol. 4, No. 1, July 2013

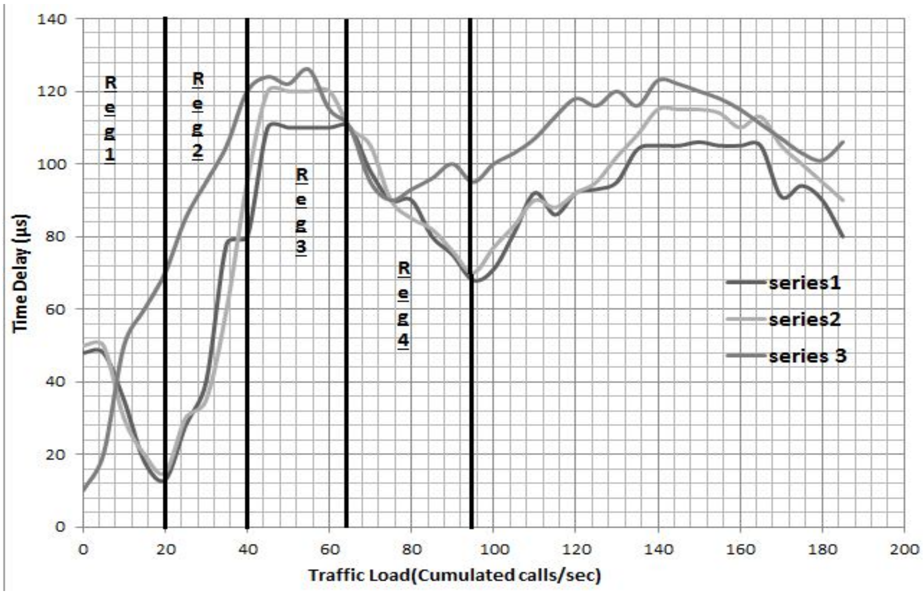

Fig. 4: Time Delay Vs Traffic Load

L_i=It is the load on that particular cluster head and can be formulated as A_i-R_i

T_t=Unit of time the cluster head is occupied with call transmission only and not computation.

So while using only static cluster-head, our overall delay equation will become:

DELAY (*D*) = {Log (n)*d} + {Log (n)-1}* *k* + WTD

DELAY (*D*) = {Log (n)*d} + {Log (n)-1}* *k* + f (Li,Tt)- - (1)

The main problem here is that the WTD delay completely depends on network traffic and we have no control over it. It may happen that the cluster head may remain busy for greater time period and may increase the delay for indefinite period of time.

Delay reduction on dynamically cluster head approach:

When dynamic approach is used along with static approach, the WTD delay will disappear, because when the cluster head becomes busy, the idlest node in the cluster is made the cluster head. But this transition of cluster head entails selection of the least loaded node and also transferring the entire set of information of the cluster from previous to current cluster head. So, all these tasks will lead a new type of delay which can be termed as Transition Delay [TD] and can be defined as:

TD=f (|C|, Li) where, |C|=cardinality of that cluster and Li is the load of the i^{th} cluster.

But this delay will only happen after T_R time interval, when this dynamic procedure will work.

So, our new delay equation after every T_R time interval will be:

DELAY (*D*) = {Log (n)*d} + {Log (n)-1}* *k* +TD

DELAY (*D*) = {Log (n)*d} + {Log (n)-1}* *k* + f (|C|) —— (2)

But rest of the time the delay will be a normal delay as discussed above:

DELAY (*D*) = [{Log (n)*d} + [{Log (n)-1}* *k*]]

So when we are using the combined dynamic and static approach rather than only static approach we only have to incur Translation delay (**TD**), that to after every T_R interval. But the indefinite Waiting Time Delay (WTD) is completely removed. As a result the delay reduces considerably when the network becomes busy or saturated.

III. Experimental Results:

The results of the proposed model have been plotted in a graph (series 1) as shown in Fig 4. Here, X-axis represents the Traffic load in terms of cumulative calls/second and the Y-axis represents the corresponding time delay (µs).It has also been compared with the results of the previous two models [2] and [4] which have been plotted in series 2 and series 3 respectively.

The plot can be subdivide into four regions as shown in Fig 4.These regions are described as below:

Reg 1: *Region of Dormancy*:

It's the pre-computation period so delay is more in series 1 and 2, compared to series 3, as the calls are made to wait till pre-computation is over completely.

Reg 2: *Incremental Region*:

As the pre-computation is already done, so the calls are accepted by the free nodes at a rate faster in series 1 and 2 than the previous model(series 3).

Reg 3: *Saturation Region*:

As the Traffic Load increases, so more nodes are being engaged in accepting calls, resulting in utilization of free resources, so in this region, the calls have to suffer an almost steady delay due to unavailability of resources.

Reg 4: *Decrementing Region*:

As the previous accepted calls release their resources after their bound time, so the availability of free nodes increases.These results in increase in call acceptance rate and hence delay decreases.

On comparing series 1 and series 2, we can see that in saturated region the delay is more in series 2 where only static selection of cluster head approach is used, than on series 1 where combined static and dynamic selection of

DOI: 01.IJNS.4.1.1160

ACEEE Int. J. on Network Security , Vol. 4, No. 1, July 2013

cluster head approach is used this is due to the absence of WTD delay. Also if we closely watch the series 1, we can see than, at some points the delay suddenly increases for very minute period of time and this is due to Transition Delay [TD] which appears after T_R interval.

IV. Conclusion

In this model, cluster heads are being dynamically chosen, so that even during heavy traffic hours, both call transmissions and ORP computation can be carried out with ease. When the cluster head is busy in transmitting calls through it, a new least loaded node is chosen to be the new cluster head. Hence, the efficiency of the model has been improved with the reduction of delay due to least probable backtracking and also by discarding the delay caused due to busy cluster-head. So, by using combined static and dynamic selection of cluster head approach the delay is considerable reduced in saturated region compared to our previous approach [2], where we have only used the static selection of cluster head approach. The experimental result of series 1, 2 and 3 proves our point.

References

[1]. P.K.Guha Thakurta and Subhansu Bandyopadhyay, "A New Dynamic Pricing Scheme with Priority based Tree Generation and Scheduling for Mobile Networks", IEEE Advanced Computing Conference, March 2009.

[2]. Subhajit Sadhu, Sushovon Maity ,Vivek Agarwal and P K Guha Thakurta,"A New Cluster Based Approach on Load Balancing Technique with Congestion Control Schema for Mobile Networks", IEEE MNCApps Conference, August 2012.

[3]. Saravut Yaipairoj and Fotios C.Harmantzis, "A Dynamic Pricing Model for Data Services in GPRS Networks", Globecom Workshops, IEEE Communication Society, 2004

[4]. Sarah Kabahuma and Olabisi E. Falowo, "Analysis of Network Operators' Revenue with a Dynamic Pricing Model Based on User Behaviour in NGWN Using JCAC", SATNAC, 2010.

[5]. P.K. GuhaThakurta, Prothoma Sinha, Nilesh Mallick and Subhansu Bandyopadhyay, "An Approach towards Reduction of Routing Paths for Mobile Networks", ICMOC, 2010.

[6]. Andrew McCallum, Kamal Nigam, Lyle H. Ungar, "Efficient Clustering of High Dimensional Data Sets with Application to Reference Matching", 6th ACM SIGKDD International Conference on Knowledge discovery and Data Mining, 2000.

[7]. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein "Introduction to Algorithms", Third Edition 2010.

DOI: 01.IJNS.4.1.1160